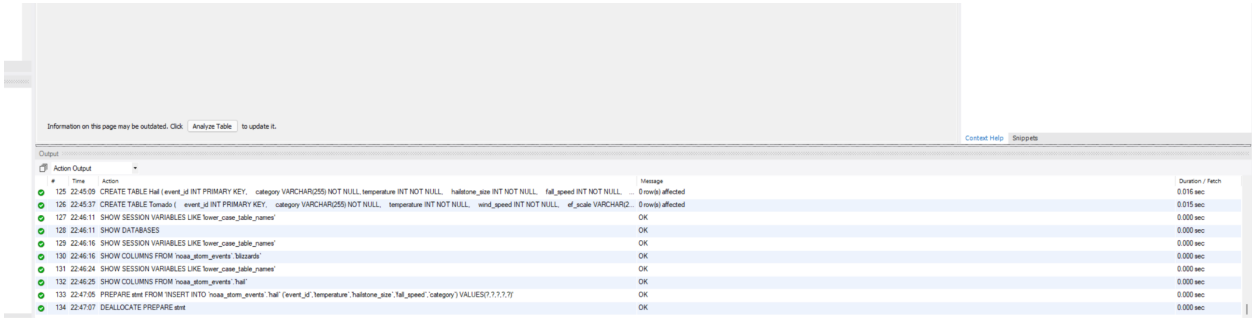
Running locally via MySQL Workbench:
Sample CSV generated with Mockaroo: https://www.mockaroo.com/ (1000 entry limit)



Setting up SQL instance on GCP:

```
meizukon@cloudshell:~ (cs411-project-400321)$ gcloud sql connect stormevent
s --user=root
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 425150
Server version: 8.0.31-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input stateme
nt.

mysql>
```

Database Design Queries:

```
CREATE TABLE User (
  User_id  INT AUTO_INCREMENT PRIMARY KEY,
  user_name VARCHAR(255),
  password VARCHAR(255)
);

CREATE TABLE Category (
  Category_id  INT PRIMARY KEY,
  category_name VARCHAR(255)
);

CREATE TABLE WeatherEvent (
  Event_id  INT AUTO_INCREMENT PRIMARY KEY,
  User_id INT,
```

```
  category_id INT,
  eventBeginTime DATETIME,
  eventEndTime DATETIME,
  damageProperty VARCHAR(255),
  place VARCHAR(255),
  deathsDirect INT,
  deathsIndirect INT,
  injuriesDirect INT,
  injuriesIndirect INT,
  damageCrops VARCHAR(255),
  FOREIGN KEY (User_id) REFERENCES User(User_id),
  FOREIGN KEY (category_id) REFERENCES Category(category_id)
);


CREATE TABLE Tornado (
  tornado_id  INT AUTO_INCREMENT PRIMARY KEY,
  event_id INT,
  temperature DECIMAL,
  wind_speed INT,
  ef_scale VARCHAR(255),
  tornado_size DECIMAL,
  FOREIGN KEY (event_id) REFERENCES WeatherEvent(event_id)
);


CREATE TABLE Blizzard (
  Blizzard_id  INT AUTO_INCREMENT PRIMARY KEY,
  event_id INT,
  temperature DECIMAL,
  wind_speed INT,
  snow_depth DECIMAL,
  FOREIGN KEY (event_id) REFERENCES WeatherEvent(event_id)
);


CREATE TABLE Hail (
  hail_id INT AUTO_INCREMENT PRIMARY KEY,
  event_id INT,
  temperature DECIMAL,
  hailstone_size DECIMAL,
  fall_speed DECIMAL,
  FOREIGN KEY (event_id) REFERENCES WeatherEvent(event_id)
);
```

```
CREATE TABLE UserFavoriteEvents (
  favorite_event_id INT AUTO_INCREMENT PRIMARY KEY,
  User_id INT,
  event_id INT,
  FOREIGN KEY (User_id) REFERENCES User(User_id),
  FOREIGN KEY (event_id) REFERENCES WeatherEvent(event_id)
);

CREATE TABLE Thunderstorm (
  thunderstorm_id INT PRIMARY KEY,
  event_id INT,
  temperature DECIMAL,
  wind_speed INT,
  FOREIGN KEY (event_id) REFERENCES WeatherEvent(event_id)
        );
```

Screenshots of at least 1000 rows per table:

```
mysql> select count(*) from Blizzard;
+----------+
| count(*) |
+----------+
|     1751 |
+----------+
1 row in set (0.00 sec)
```

select count(*) from Category;

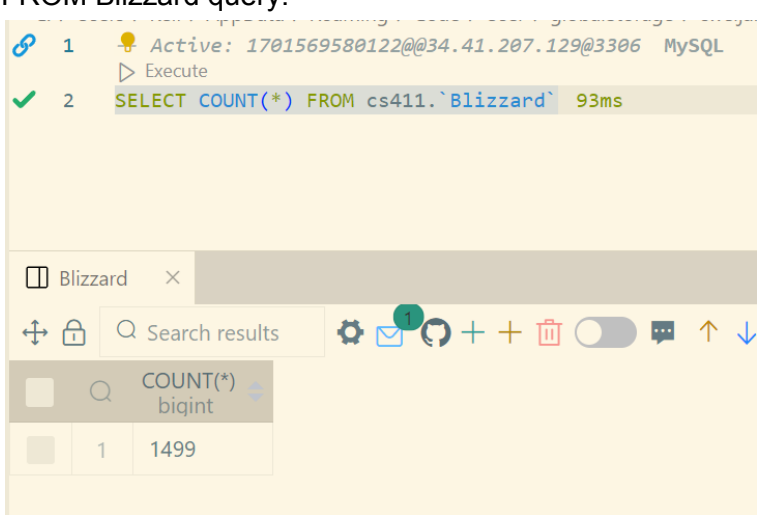
```
mysql> select count(*) from Hail;
+----------+
| count(*) |
+----------+
|     2094 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> select count(*) from Tornado;
+----------+
| count(*) |
+----------+
|     1756 |
+----------+
1 row in set (0.00 sec)
```

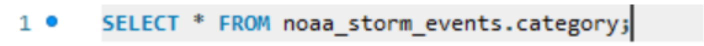
select count(*) from User;

```
mysql> select count(*) from WeatherEvent;
+----------+
| count(*) |
+----------+
|     6306 |
+----------+
1 row in set (0.00 sec)
```

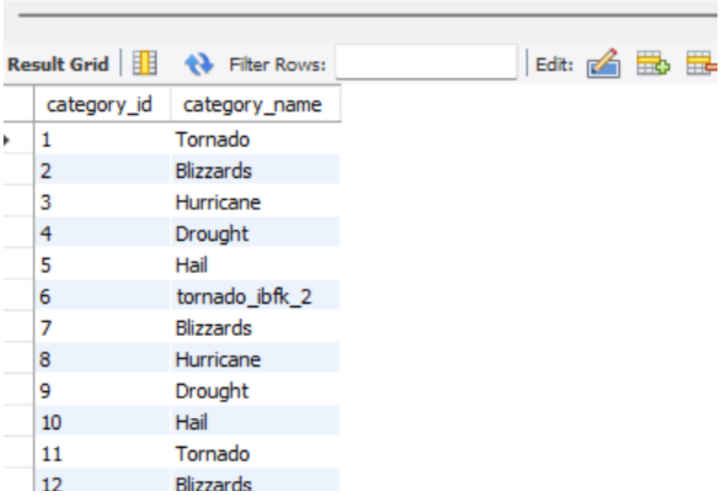
Sample select count (*)
FROM Blizzard query:



Sample select * from categories query:



```sql
SELECT * FROM noaa_storm_events.category;
```

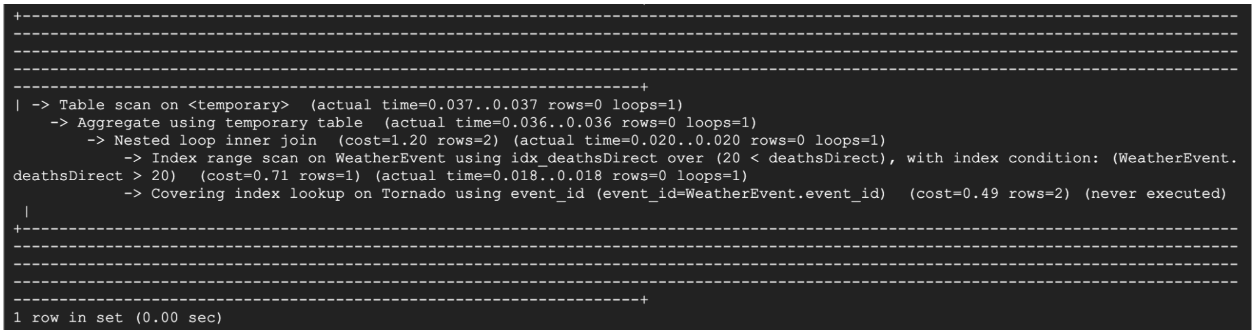


category_id	category_name
1	Tornado
2	Blizzards
3	Hurricane
4	Drought
5	Hail
6	tornado_ibfk_2
7	Blizzards
8	Hurricane
9	Drought
10	Hail
11	Tornado
12	Blizzards

Complex Queries:

Selection of the count of tornadoes directly resulting in a death toll above 20 per category of tornado.

SELECT count(*)
FROM Tornado JOIN WeatherEvent on (Tornado.event_id = WeatherEvent.event_id)
WHERE deathsDirect > 20
GROUP BY category_id;



```
+------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.037..0.037 rows=0 loops=1)
    -> Aggregate using temporary table  (actual time=0.036..0.036 rows=0 loops=1)
        -> Nested loop inner join  (cost=1.20 rows=2) (actual time=0.020..0.020 rows=0 loops=1)
            -> Index range scan on WeatherEvent using idx_deathsDirect over (20 < deathsDirect), with index condition: (WeatherEvent.
deathsDirect > 20)  (cost=0.71 rows=1) (actual time=0.018..0.018 rows=0 loops=1)
            -> Covering index lookup on Tornado using event_id (event_id=WeatherEvent.event_id)  (cost=0.49 rows=2) (never executed)
 |
+------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Selection of the size of hailstone typical of a hail storm resulting in less than 50 or greater than 70 direct deaths.

```
SELECT hailstone_size FROM Hail
WHERE EXISTS (
   SELECT *
   FROM Hail
   JOIN WeatherEvent ON Hail.event_id = WeatherEvent.event_id
   WHERE (WeatherEvent.deathsDirect < 50 AND Hail.hailstone_size IS NOT NULL)
      OR (WeatherEvent.deathsDirect > 70 AND Hail.hailstone_size IS NOT NULL)
LIMIT 15;
);
```

```
+----------------+
| hailstone_size |
+----------------+
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
|              3 |
+----------------+
15 rows in set (0.00 sec)
```

```
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------+
| -> Limit: 15 row(s)  (actual time=1.712..1.713 rows=15 loops=1)
    -> Sort: Hail.hailstone_size, limit input to 15 row(s) per chunk  (actual time=1.710..1.711 rows=15 lo
ops=1)
        -> Stream results  (cost=693.74 rows=1696) (actual time=0.943..1.502 rows=651 loops=1)
            -> Inner hash join (no condition)  (cost=693.74 rows=1696) (actual time=0.940..1.438 rows=651
loops=1)
                -> Filter: (Hail.hailstone_size is not null)  (cost=211.15 rows=1885) (actual time=0.811..
1.214 rows=651 loops=1)
                    -> Table scan on Hail  (cost=211.15 rows=2094) (actual time=0.009..1.103 rows=2094 loo
ps=1)
                -> Hash
                    -> Limit: 1 row(s)  (cost=482.58 rows=1) (actual time=0.114..0.114 rows=1 loops=1)
                        -> Nested loop inner join  (cost=482.58 rows=1152) (actual time=0.113..0.113 rows=
1 loops=1)
                            -> Filter: ((WeatherEvent.deathsDirect < 50) or (WeatherEvent.deathsDirect > 7
0))  (cost=75.45 rows=412) (actual time=0.073..0.073 rows=1 loops=1)
                                -> Table scan on WeatherEvent  (cost=75.45 rows=742) (actual time=0.071..0
.071 rows=1 loops=1)
                            -> Filter: (((WeatherEvent.deathsDirect < 50) and (Hail.hailstone_size is not
null)) or ((WeatherEvent.deathsDirect > 70) and (Hail.hailstone_size is not null)))  (cost=813.27 rows=3)
(actual time=0.039..0.039 rows=1 loops=1)
                                -> Index lookup on Hail using Hail_ibfk_1 (event_id=WeatherEvent.event_id)
  (cost=813.27 rows=3) (actual time=0.037..0.037 rows=1 loops=1)
 |
+-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
----------------------------------+
| -> Limit: 15 row(s)  (actual time=0.915..0.916 rows=15 loops=1)
    -> Sort: Hail.hailstone_size, limit input to 15 row(s) per chunk  (actual time=0.914..0.915 rows=15 loops=1)
        -> Stream results  (cost=431.07 rows=651) (actual time=0.138..0.760 rows=651 loops=1)
            -> Inner hash join (no condition)  (cost=431.07 rows=651) (actual time=0.135..0.670 rows=651 loops=1)
                -> Filter: (Hail.hailstone_size is not null)  (cost=131.43 rows=651) (actual time=0.007..0.411 rows=651 loops=1)
                    -> Covering index range scan on Hail using idx_hailstone_size over (NULL < hailstone_size)  (cost=131.43 rows=651
) (actual time=0.006..0.362 rows=651 loops=1)
                -> Hash
                    -> Limit: 1 row(s)  (cost=299.64 rows=1) (actual time=0.117..0.118 rows=1 loops=1)
                        -> Nested loop inner join  (cost=299.64 rows=221) (actual time=0.116..0.116 rows=1 loops=1)
                            -> Filter: ((WeatherEvent.deathsDirect < 50) or (WeatherEvent.deathsDirect > 70))  (cost=50.73 rows=252)
(actual time=0.089..0.089 rows=1 loops=1)
                                -> Covering index range scan on WeatherEvent using idx_deathsDirect over (NULL < deathsDirect < 50) O
R (70 < deathsDirect)  (cost=50.73 rows=252) (actual time=0.020..0.020 rows=1 loops=1)
                            -> Filter: (((WeatherEvent.deathsDirect < 50) and (Hail.hailstone_size is not null)) or ((WeatherEvent.de
athsDirect > 70) and (Hail.hailstone_size is not null)))  (cost=156.06 rows=1) (actual time=0.026..0.026 rows=1 loops=1)
                                -> Index lookup on Hail using idx_event_id_Hail (event_id=WeatherEvent.event_id)  (cost=156.06 rows=3
) (actual time=0.025..0.025 rows=1 loops=1)
 |
+-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
```

Advanced Subqueries

Join and Aggregation: Find the total number of direct and indirect deaths for each category of weather events.

```
SELECT Category.category_name,
    SUM(WeatherEvent.deathsDirect) AS TotalDirectDeaths,
    SUM(WeatherEvent.deathsIndirect) AS TotalIndirectDeaths
FROM WeatherEvent
JOIN Category ON WeatherEvent.category_id = Category.category_id
GROUP BY Category.category_name;
```

```
mysql> SELECT Category.category_name,
    ->        SUM(WeatherEvent.deathsDirect) AS TotalDirectDeaths,
    ->        SUM(WeatherEvent.deathsIndirect) AS TotalIndirectDeaths
    -> FROM WeatherEvent
    -> JOIN Category ON WeatherEvent.category_id = Category.category_id
    -> GROUP BY Category.category_name;
+---------------+-------------------+---------------------+
| category_name | TotalDirectDeaths | TotalIndirectDeaths |
+---------------+-------------------+---------------------+
| Tornado       |               217 |                 130 |
| Blizzard      |               243 |                 210 |
| Hail          |               225 |                 102 |
| Thunderstorm  |               216 |                 123 |
+---------------+-------------------+---------------------+
4 rows in set (0.01 sec)

mysql>
```

Subquery and Join: List the top 5 largest hailstone sizes recorded and their associated weather event details.

SELECT WE.event_id, WE.eventBeginTime, WE.eventEndTime, H.hailstone_size
FROM WeatherEvent WE
JOIN Hail H ON WE.event_id = H.event_id
JOIN (SELECT hailstone_size FROM Hail ORDER BY hailstone_size DESC LIMIT 5) AS TopHail ON H.hailstone_size = TopHail.hailstone_size;

```
mysql> SELECT WE.event_id, WE.eventBeginTime, WE.eventEndTime, H.hailstone_size
    -> FROM WeatherEvent WE
    -> JOIN Hail H ON WE.event_id = H.event_id
    -> JOIN (SELECT hailstone_size FROM Hail ORDER BY hailstone_size DESC LIMIT 5) AS TopHail
ON H.hailstone_size = TopHail.hailstone_size;
+----------+---------------------+---------------------+----------------+
| event_id | eventBeginTime      | eventEndTime        | hailstone_size |
+----------+---------------------+---------------------+----------------+
|   927672 | 2023-12-06 07:04:52 | 2023-12-07 07:04:52 |              4 |
|   927672 | 2023-12-06 07:04:52 | 2023-12-07 07:04:52 |              4 |
|   927672 | 2023-12-06 07:04:52 | 2023-12-07 07:04:52 |              4 |
|   423611 | 2023-12-06 07:04:52 | 2023-12-06 12:04:52 |              4 |
|   423611 | 2023-12-06 07:04:52 | 2023-12-06 12:04:52 |              4 |
|   423611 | 2023-12-06 07:04:52 | 2023-12-06 12:04:52 |              4 |
|   607507 | 2023-12-06 07:04:52 | 2023-12-06 17:04:52 |              5 |
|   607507 | 2023-12-06 07:04:52 | 2023-12-06 17:04:52 |              5 |
|   971986 | 2023-12-06 07:04:52 | 2023-12-07 01:04:52 |              4 |
|   971986 | 2023-12-06 07:04:52 | 2023-12-07 01:04:52 |              4 |
|   971986 | 2023-12-06 07:04:52 | 2023-12-07 01:04:52 |              4 |
|   610792 | 2023-12-06 07:04:52 | 2023-12-06 11:04:52 |              4 |
|   610792 | 2023-12-06 07:04:52 | 2023-12-06 11:04:52 |              4 |
|   610792 | 2023-12-06 07:04:52 | 2023-12-06 11:04:52 |              4 |
|   789518 | 2023-12-06 07:04:52 | 2023-12-06 20:04:52 |              5 |
|   789518 | 2023-12-06 07:04:52 | 2023-12-06 20:04:52 |              5 |
+----------+---------------------+---------------------+----------------+
16 rows in set (0.00 sec)
```

Set Operation (UNION) and Aggregation: Combine and count the total number of tornadoes and blizzards by state (assuming state information is in the `WeatherEvent` table).

```
SELECT WE.User_id AS State, 'Tornado' AS EventType, COUNT(*) AS TotalEvents
FROM WeatherEvent WE
JOIN Tornado T ON WE.event_id = T.event_id
GROUP BY WE.User_id
UNION
SELECT WE.User_id AS State, 'Blizzard' AS EventType, COUNT(*) AS TotalEvents
FROM WeatherEvent WE
JOIN Blizzard B ON WE.event_id = B.event_id
GROUP BY WE.User_id;
```

```
Database changed
mysql> SELECT WE.User_id AS State, 'Tornado' AS EventType, COUNT(*) AS TotalEvents
    -> FROM WeatherEvent WE
    -> JOIN Tornado T ON WE.event_id = T.event_id
    -> GROUP BY WE.User_id
    -> UNION
    -> SELECT WE.User_id AS State, 'Blizzard' AS EventType, COUNT(*) AS TotalEvents
    -> FROM WeatherEvent WE
    -> JOIN Blizzard B ON WE.event_id = B.event_id
    -> GROUP BY WE.User_id;
+-------+-----------+-------------+
| State | EventType | TotalEvents |
+-------+-----------+-------------+
|     3 | Tornado   |           3 |
|    10 | Tornado   |           2 |
|     9 | Tornado   |           2 |
|     2 | Tornado   |           3 |
|     8 | Tornado   |           5 |
|     1 | Tornado   |           2 |
|     4 | Tornado   |           1 |
|     5 | Tornado   |           1 |
|  NULL | Tornado   |         972 |
|     1 | Blizzard  |           3 |
|     5 | Blizzard  |           2 |
|     3 | Blizzard  |           3 |
|    10 | Blizzard  |           4 |
|     8 | Blizzard  |           4 |
|     9 | Blizzard  |           1 |
|     4 | Blizzard  |           4 |
|     2 | Blizzard  |           1 |
+-------+-----------+-------------+
17 rows in set (0.00 sec)
```

Join and Subquery: Find the average wind speed for tornadoes that occurred during the top 10 most damaging weather events.

SELECT AVG(T.wind_speed) AS AverageWindSpeed
FROM Tornado T
JOIN WeatherEvent WE ON T.event_id = WE.event_id
JOIN (SELECT event_id FROM WeatherEvent ORDER BY damageProperty DESC LIMIT 10)
AS TopEvents ON WE.event_id = TopEvents.event_id;

```
EXPLAIN
-> Aggregate: avg(T.wind_speed)  (cost=93.22 rows=1) (actual time=1.080..1.081 rows=1 loops=1)     -> Nested loop inner join  (cost=90.86 rows=24) (actual time=0.987..1.067 rows=10 loops=1)        -> Nested loop inner join  (cost=82.58 rows=10) (actual ...
```

EXPLAIN
-> Aggregate: avg(T.wind_speed) (cost=93.22 rows=1) (actual time=0.542..0.542 rows=1 loops=1)
-> Nested loop inner join (cost=90.86 rows=24) (actual time=0.456..0.535 rows=10 loops=1)
-> Nested loop inner join (cost=82.58 rows=10) (actual time=0.438..0.455 rows=10 loops=1)
-> Table scan on TopEvents (cost=76.71..79.08 rows=10) (actual time=0.425..0.426 rows=10 loops=1)
-> Materialize (cost=76.45..76.45 rows=10) (actual time=0.424..0.424 rows=10 loops=1)
-> Limit: 10 row(s) (cost=75.45 rows=10) (actual time=0.409..0.410 rows=10 loops=1)
-> Sort: WeatherEvent.damageProperty DESC, limit input to 10 row(s) per chunk (cost=75.45 rows=742) (actual time=0.408..0.409 rows=10 loops=1)
-> Table scan on WeatherEvent (cost=75.45 rows=742) (actual time=0.064..0.270 rows=742 loops=1)
-> Single-row covering index lookup on WE using PRIMARY (event_id=TopEvents.event_id) (cost=0.26 rows=1) (actual time=0.003..0.003 rows=1 loops=10)
-> Index lookup on T using event_id (event_id=TopEvents.event_id) (cost=0.62 rows=2) (actual time=0.007..0.008 rows=1 loops=10)